




# CF DevOps Best Practices

sharing what we learn running and seeing others run large installations of CF

IBM CF Community
Pivotal CF DevOps
contact: maxim@us.ibm.com

**v0.2.0**
April 2nd, 2015

# agenda

- preamble
- best practice **template**
- best practices
  - DevOps
  - BOSH
  - Team
- what next?

# preamble

- DevOps is **ongoing set of activities** for cloud systems
  - running, updating, managing
  - maintaining overall “health”
- not much best practices being shared today, let’s change that
- simple start: interview **Tony Hansmann** (lead DevOps @ Pivotal)
- this is a synthesis of the advices he shared with me

# best practice template or pattern

- **name** and **short description**
- summary **diagram** [optional]
- **advantages / disadvantages** [optional]
  - **when** to use
  - **when not** to use
- **who** is affected

# DevOps best practice: **github**

- use git and github (or similar tool) for all bit deployment artifacts

- main goals and advantages
  - versioning, replicating, portability, recreating, …
  - managing manifest templates for each env (for instance)
  - managing contributions from all in single place

- everyone in team should be doing this

# DevOps best practice: **checklists**

- every part of team process should be written and use a checklist for execution
- checklists provide
  - "brain-dead" recipe for all to follow
  - easy to follow by both experience team members and newbies
  - can be improved and refined over time (evolve)
  - should include as last item: create new checklist (from this) for next use
- everyone in team

# DevOps best practice: **deployment drills**

- setup drill deployments with sometimes known issues. Goal is to test team's ability to react to issues and follow process. Helps solidify/refine new changes to process before using in real situation

- primarily done to verify
  - team's preparedness
  - run through new checklists and changes thereof
  - help in on boarding new team members for what's to come

- everyone in team

# DevOps best practice: **blue/green deployments**

- common practice of using multiple staged deployment environments before changing production environment

- well known practice, adjust
  - ratio of non-production to production envs (2:1 is common)
  - systematic testing of new releases
  - replicating workloads, Pivotal's **Tabasco** & **A1**

- planning team, management team

# DevOps best practice: **canary deployments**

- always use canary deployments… tells you like (real canaries) that something maybe wrong before going further with deployment
- some things to consider
  - smoke tests are good complements
  - monitoring canaries (datadog, New Relic)
- team leads

# DevOps best practice: **monitoring dashboards**

- Pivotal DevOps operation rely heavily on collected real time metrics and graphs to help decisions and knowing the state of each deployment
- some suggestions
  - replicate same graphs as Pivotal
  - customize to address your needs
  - NewRelic is great tool that is used heavily
  - Datadog and Pingdom ( on-call teams)
  - Splunk and others such tools for systematic processing and investigating logs
- team leads

# DevOps best practice: **max in flight**

- set to allow rolling deployments
- empirical evidence for value
  - max in flight setting (based on RAM for fleet)

    (Free RAM / 2 ) / RAM per DEA = # DEA for rolling deployment
  - play around with formulae to adjust to your needs
  - always use rolling deployments for your production ends
- team leads

# BOSH best practice: **µBOSH**

- use micro-BOSH (µBOSH) as bootstrap to all BOSH deployments
- main advantages
  - can manage multiple deploys
  - allows update to director for each deploy
  - another layer of redundancy when things go wrong
  - new µBOSH CLI helps streamline process or creating and managing µBOSH
- disadvantages: adds some complexity to most of the processes
- everyone

# BOSH best practice: **jumpbox**

- use jumpbox VM as your main entry to your production environment
- advantages
  - single point of access control
  - can be redeployed often to update and change keys
  - bridges your network with the environment's
  - share keys to env to all, but keys jumpbox to who need it
- planning team, DevOps teams, and managers

# BOSH best practice: **resurrector**

- BOSH will help resurrect VMs and jobs that it knows are failing or have failed. Use `$bosh cck`
- advantages
  - resurrector can “replay” packages setup and jobs to bring back VM to known good states
  - easy to use and keeps the BOSH director DB consistent
  - various options provided: restart jobs, recreate VMs, delete VMs, etc.
- disadvantages: thorough and slow, and sometimes does not work (force recreate VM option is last resort)
- DevOps teams and on call teams

# BOSH best practice: **manifest maintenance**

- manifest is the source of truth for each of your deployment. Track all changes on github
- advantages
  - small changes can be tracked and rolled back
  - always have a way to recreate a version of your environment
  - use diffs to manage complexity of manifest changes
  - start new deployment with previous deployment's manifest and modify
- everyone

# BOSH best practice: **use stock BOSH**

- Pivotal DevOps team uses stock BOSH for all deployments. No changes to the code. No special plugins or no special stemcells
- advantages
  - no need to keep track of additional repos
  - no need to update BOSH releases or stemcells
  - no need to update any parts of BOSH
  - no magic
- lead and management team need to decide… sometimes not possible

# BOSH best practice: **stemcell hygiene**

- trying to stay up to date on latest stemcells often avoids the major (error prone) BIG update in future
- advantages
  - helps with team's cadence
  - means DEAs are rebooted which can also help with their health
  - less worry about security since stemcells contain updated OS and packages which is the main source of security fixes for known vulnerabilities
- lead and management team

# Team best practice: **on boarding newbies**

- all team members pair. On-boarding amounts to having experience team pair with newbies. Newbies need to experience new BOSH deploy from scratch (e.g., dummy deploy and dummy-with-package deploy)

- advantages
  - sharing of knowledge
  - experience debugging deploys
  - experience using checklists
  - improve on boarding with each newbie (use an on boarding checklist)

- lead and new team members

# Team best practice: **experiencing debug pain**

- learn how to debug a failing CF application, CF service, or DEA. Learn the BOSH commands for debugging, e.g., `debug` and `log` commands and `ssh` to access VMs

- advantages
  - every team member must know how to debug
  - only through experience can one really learn and get a feel for what is wrong with an environment
  - debugging checklist can be useful and necessary, but really just a guide

- everyone

# Team best practice: **challenging team members**

- checklist-driven DevOps can be boring. Make sure to challenge team members. Rotate pairs and responsibilities

- advantages

  - share tacit knowledge across team and sub-teams
  - new pair of eyes on old problems
  - more energized team members in long run
  - keep folks happy, avoids boredom

- everyone

# next steps

- **refine** template lists
- **collect more** best practices
  - interview other Pivotal DevOps team
  - interview Bluemix DevOps team
- **create collective repository** of best practices?
- share, refine… rinse, and repeat…

# backup

backup

# typical BOSH deploy